certain modules for capturing and logging different types of computer activity in a normalized format to more easily facilitate advanced analysis and data reduction techniques. An application that runs on each system and captures various user and software activities on that system while logging them to a database for future analysis.

This will enable forensic investigators to collect computer activity data samples from Linux operating system. These details can then be investigated for research. The traditional computer forensics investigative methodology is both time taking and costly in terms of human resources and laboratory software and hardware [7]. The acquisition of data from storage media post-mortem poses critical efficiency problems for investigators, as the size of computer storage media increases corresponding with an expansion of computer related activities. So the proposed way seems to be helpful.

Mainly these types of data are collected by the system. These are organised into modules and they are,

- File system event watcher: It records the activity taking place in the file system.
- Removable device discovery: It records whether any external devices are plugged into the system.

In the proposed system each module carries out a specific function. Each module collects events and stores it into a database. The output is saved so as to enable further analysis. The database contains the details about the file system events happened in a system, and USB connection details. These details stored in the database can be retrieved by an investigator by creating an interface.

2.1 File system event watcher

File system is responsible for storing information on disk and retrieving and updating this information. In Linux all the things are treated as a file. When dealing with Linux machines, Ext based file systems will be found frequently.Ext3 is an extension of ext2 that added journaling to the mix to relieve us of those horribly long file system checks when something has gone awry. Ext4 is a deeper enhancement over Ext3 than Ext3 was over Ext2.

The need to scan a given file system for changes is a fairly common one, and there are a number of common tasks which need this, including:

- Notifying applications of alterations in configuration files.
- Tracking modifications in critical system files.

A common approach to doing this sort of change notification is file polling, however this appears to be ineffective for all but the most frequently-changed files and can miss some types of changes. Data integrity systems like Tripwire track file changes based on a fixed time schedule, but the time-scheduled method doesn't work if you want to be notified every time it changes in real-time just as an event takes place. A framework which fulfills that requirement is Inotify. Here also we are using this functionality.

Inotify is a file change notification system in the Linux kernel, accessible since version 2.6.13. [8]. It is an efficient way to trace actions in the file system on Linux in real-time. The inotify API provides a method for monitoring file system events. It can be used to monitor individual files or to monitor directories. When a directory is noticed inotify will give events for the directory itself, and files inside the directory. It replaces an earlier facility called dnotify which had similar goals. One major use is in desktop search utilities where its functionality allows re indexing of changed file system for changes every few minutes, which would be very inefficient. The kernel directly told that a file has changed. But inotify does not support repeatedly watching directories, meaning that a different inotify watch must be created for each sub directory.

2.1.1 Inotify Execution flow

- Creating inotify monitoring (watch) list. Add the directories/files to the inotify monitoring (watch) list that you want to watch. We can change the monitoring list as and when needed.
- Request Inotify to report particular event changes to the monitoring list of files and directories. For example, we can request inotify to perform checking and report events such as ON ACCESS, ON OPEN, ON WRITING, ON CLOSE etc.,

2.1.2 Used functions

- Creating the inotify instance by inotify_init().The more recent inotify_init1() is like inotify_init() but it will provide some extra service [9].
- Adding all the directories to be monitored to the inotify list using inotify_add_watch() function. This function manipulates the watch list associated with an inotify instance. Each item (“watch”) in the watch list specifies the path name of a file or directory with some set of events that the kernel should watch or monitor for the file referred to by that pathname.
- To determine the changes or events happened, perform read() on the inotify instance. This read will get blocked until the change event occurs. Read returns list of events happened on the monitored directories. On the basis of the return value of read(), we will know precisely what kind of changes occurred.
- In case of separating or removing the watch on directories/files, call inotify_rm_watch().

2.1.3 Events

- IN_ACCESS – File was accessed[9]
- IN_ATTRIB – Metadata (data that describes other data) changed (Permissions, timestamps, extended attributes, etc.)
- IN_CLOSE_WRITE – It means that the file opened for writing was closed
- IN_CLOSE_NOWRITE – This event refers that, file not opened for writing was closed
- IN_CREATE – File/directory was created in watched directory
- IN_DELETE – File/directory was deleted from watched directory

- IN_DELETE_SELF – Watched file/directory was itself deleted. This event also occurs if an object is moved to another filesystem ,since in effect copies the file to the other filesystem and then deletes it from the original filesystem
- IN_MODIFY – It means that the file was modified
- IN_MOVE_SELF – Watched file/directory was itself moved
- IN_MOVED_FROM – A file was moved out of watched directory
- IN_MOVED_TO – File moved into watched directory
- IN_OPEN –File was opened

3. Conclusion

A computer-related investigation often produces a particularly large volume of evidence [11]. Managing all this data and using it effectively throughout the span of an investigation presents special problems. The proposed system will log certain activities taking place in a computer system that may help forensic examiners. The proposed system is different than other existing logging systems as it captures real-time activities taken by the users such as adding, deleting, or modifying a file and connecting a USB device.

4. Acknowledgement

We would like to thank, first and foremost, Almighty God, without his support this work would not have been possible. We would also like to thank all the faculty members of College of Engineering Kallooppara for their immense support

References

[1] Ubuntu manual team, Getting started with Ubuntu 14.04
[2] https://en.wikipedia.org/wiki/Inode
[3] Linux Forensics (for Non -Linux Folks) Hal Pomeranz Deer Run Associates
[4] Network Security Bible Second Edition, Eric Cole
[5] Unix and Linux forensic analysis DVD toolkit, Chris Pogue, Cory Altheide, Todd Haverkos
[6] Kenneth E. Nawyn, " A Security Analysis of System Event Logging with Syslog", SANS Institute, 2003.
[7] Shadi Al Awawdeh, Ibrahim Baggili, Andrew Marrington, FarkhundIqbal, "CAT Record (Computer Activity Timeline Record): A unified agent based approach for real time computer forensic evidence collection", IEEE 2013.
[8] http://people.clarkson.edu/
[9] http://www.thegeekstuff.com/
[10] http://askubuntu.com/
[11] https://www.fbi.gov/

Author Profile

Neethu P Nair obtained B.Tech in Computer Science and Engineering from Mount Zion College of Engineering, Kadammanitta in 2013. She is now pursuing her master degree in Computer Science with specialization in Cyber Forensics and Information Security at College of Engineering, Kallooppara under Cochin University of Science and Technology.

Maniveena C obtained B.Tech and M.Tech in Computer Science and Engineering from Anna University. She is currently working as Assistant Professor in College of Engineering, Kallooppara.